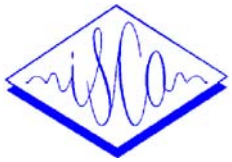

INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

# COST-SENSITIVE CALL CLASSIFICATION

*Gokhan Tur*

AT&T Labs-Research,
Florham Park, NJ, 07932, USA
gtur@research.att.com

## ABSTRACT

We present an efficient and effective method which extends the Boosting family of classifiers to allow the weighted classes. Typically classifiers do not treat individual classes separately. For most real world applications, this is not the case, not all classes have the same importance. The accuracy of a particular class can be more critical than others. In this paper we extend the mathematical formulation for Boosting to weigh the classes differently during training. We have evaluated this method for call classification in AT&T spoken language understanding system. Our results indicate significant improvements in the "important" classes without a significant loss in the overall performance.

## 1. INTRODUCTION

Classification algorithms have long been studied in the machine learning community. Typically the proposed classification algorithms do not treat individual classes separately for multi-class classification. The benchmark results are presented either for binary classification case for each class or a multi classification case where all classes have the same importance or cost. For most real world applications, this is not the case, not all classes have the same weight. The accuracy of a particular class can be more critical than others or high precision may be needed for some classes. Especially while dealing with more than a few number of classes, this is unavoidable.

Our domain is call classification where we aim to route the input calls in a customer care call center [1]. In this application, callers are greeted by the open ended prompt "*How May I Help You?*." Then the system tries to identify the customer's intent (call-type), which is basically framed as a call classification problem. In the event the system is unable to understand the caller then the conversation will proceed with either a clarification or a confirmation prompt. As a call classification example, consider the utterance *I would like to know my account balance*. Assuming that the utterance is recognized correctly, the corresponding call-type would be *Tellme(Balance)* and the action would be prompting the balance to the user or routing this call to the billing department. In such a system some call-types may require higher precision than others. According to our knowledge none of the previous call classification systems [1, 2, 3, among others] consider cost-sensitive classification.

In this paper we propose an effective and efficient cost sensitive learning approach for Boosting family of classifiers [4]. We extend the mathematical formulation for Boosting to weigh the classes differently during training. Boosting is an iterative algorithm, and at each iteration a weak learner, which minimizes an error function is determined. Our main idea is to change that error function considering the weights of the classes, thus ensuring optimal accuracy at each iteration for the "important" classes.

In the literature cost-sensitive learning is studied both for individual [5, among others] and for any [6, 7, among others] classification algorithms. Fan *et al*. have proposed the AdaCost algorithm which extends the AdaBoost algorithm giving weights to individual training examples instead of classes [5].

We briefly describe Boosting in the next section. Section 3 presents our method, and in Section 4 we show our results using this updated formulation using a call classification task within AT&T spoken dialog system.

## 2. BOOSTING

Boosting aims to combine "weak" base classifiers to come up with a "strong" classifier [4]. This is an iterative algorithm, and in each iteration, a weak classifier is learned so as to minimize the training error.

The algorithm generalized for multi-class and multi-label classification is given in Figure 1. Let $\mathcal{X}$ denote the domain of possible training examples and let $\mathcal{Y}$ be a finite set of classes of size $|\mathcal{Y}| = k$. For $Y \subseteq \mathcal{Y}$, let $Y[l]$ for $l \in \mathcal{Y}$ to be

$$Y[l] = \begin{cases} +1 \; if \; l \in Y \\ -1 \; if \; l \notin Y \end{cases}$$

The algorithm begins by initializing a uniform distibution, $D_1(i, l)$, over training examples, $i$, and labels, $l$. After each round this distribution is updated so that the example-class combinations which is easier to classify (the ones that

- Given the training data from the instance space $X$: $(x_1, Y_1), ..., (x_m, Y_m)$ where $x_i \in \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$
- Initialize the distribution $D_1(i,l) = \frac{1}{mk}$
- For each iteration $t = 1, ..., T$ do
  - Train a base learner, $h_t$, using distribution $D_t$.
  - Update

$$D_{t+1}(i,l) = \frac{D_t(i,l)e^{-\alpha_t Y_i[l]h_t(x_i,l)}}{Z_t}$$

    where $Z_t$ is a normalization factor and $\alpha_t$ is the weight of the base learner.
- Then the output of the final classifier is defined as:

$$H(x,l) = sign(f(x,l))$$

where $$f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$$

**Fig. 1**. The algorithm *Adaboost.MH*.

are already classified correctly) get lower weights and vice versa. The intended effect is to force the weak learning algorithm to concentrate on examples and labels that will be the most beneficial to the overall goal of finding a highly accurate classification rule.

## 3. APPROACH

Schapire and Singer [8] have proved a bound on the empirical *Hamming loss* (HL) of $H$ in the Boosting algorithm.

$$HL(H) \leq \prod_{t=1}^{T} Z_t$$

where $Z_t$ is the normalization factor computed on round $t$:

$$Z_t = \sum_{l \in \mathcal{Y}} \sum_{i=1}^{m} D_t(i,l)e^{-\alpha_t Y_i[l]h_t(x_i,l)}$$

Hamming loss is defined as the fraction of examples, $i$, and labels, $l$, for which the sign of $f(x_i, l)$ is different than $Y_i[l]$:

$$HL(H) = \frac{1}{mk}|i,l : H(x_i,l) \neq Y_i[l]|$$

$$= \frac{1}{mk} \sum_l \sum_i \delta(x_i,l) \quad (1)$$

where

$$\delta(x_i,l) = \begin{cases} 1 \; if \; H(x_i,l) = Y_i[l] \\ 0 \; otherwise \end{cases}$$

The proof is as follows:
By unraveling the update rule, we have that

$$D_{T+1}(i+1) = \frac{e^{-Y_i[l]f(x_i,l)}}{mk \prod_t Z_t} \quad (2)$$

Moreover, if $H(x_i,l) \neq Y_i[l]$ then $Y_i[l]f(x_i,l) \leq 0$ implying that $e^{-Y_i[l]f(x_i,l)} \geq 1$. Thus,

$$\delta(x_i,l) \leq e^{-Y_i[l]f(x_i,l)} \quad (3)$$

Combining Equations 1, 2, and 3, we get

$$\frac{1}{mk} \sum_l \sum_i \delta(x_i,l) \leq \frac{1}{mk} \sum_l \sum_i e^{-Y_i[l]f(x_i,l)}$$

$$= \sum_l \sum_i \left( \prod_{t=1}^{T} Z_t \right) D_{T+1}(i,l) = \prod_{t=1}^{T} Z_t \quad \blacksquare$$

This bound is important, because in order to minimize this training error, a reasonable approach would be minimizing $Z_t$ on each round of boosting. This leads to criteria for finding weak hypotheses, $h_t(x,l)$ for a given iteration, $t$. Assume that the weak learners, $h$, make their predictions based on a partitioning of the domain $\mathcal{X}$ into disjoint blocks $X_j$. Let $c_{jl} = h(x,l)$ for $x \in X_j$. Define

$$W_b^{jl} = \sum_{i:x_i \in X_j} D(i,l)(1 - \delta(x_i,l))$$

Using this terminology:

$$Z_t = \sum_l \sum_j \sum_{i:x_i \in X_j} D_t(i,l)e^{-Y_i[l]c_{jl}}$$

$$= \sum_l \sum_j W_+^{jl} e^{-c_{jl}} + W_-^{jl} e^{c_{jl}}$$

It is then straightforward to see that the optimal $c_{jl}$, minimizing $Z_t$ is:

$$c_{jl} = \frac{1}{2} ln \left( \frac{W_+^{jl}}{W_-^{jl}} \right)$$

Putting this in place results in:

$$Z_t = \sum_l \sum_j 2\sqrt{W_+^{jl} W_-^{jl}}$$

Then it is enough to choose the weak learner which minimizes this value.

Assume that not all labels are equally important, i.e. there is an associated cost or weight, $w_l$, to a given label, $l$. In such a case we need to define a weighted Hamming loss (WHL):

$$WHL(H) = \frac{1}{mk} \sum_l w_l \sum_i \delta(x_i, l) \quad (4)$$

It is easy to see that the above proof does not hold for $w_l > 1$. Thus, in order to get a weak classifier at each iteration, it may not be enough just to minimize $Z_t$. Even if one restricts the class weights to be less than 1, $w_l < 1$, it may be possible to find a better function minimize other than simply $Z_t$.

THEOREM 1. Assuming the notation of Figure 1, and weights of the classes, $w_l$, the following bound holds on the weighted Hamming loss of $H$:

$$WHL(H) \leq \prod_{t=1}^{T} Y_t \;\; if \; w_l \geq 1 \; \forall \, l$$

where $$Y_t = \sum_{l \in \mathcal{Y}} \hat{w}_l \sum_{i=1}^{m} D_t(i,l) e^{-\alpha_t Y_i[l] h_t(x_i, l)}$$

where $\hat{w}_l$ is a function of $w_l$.

**Proof:** By unraveling the update rule, we have that

$$D_{T+1}(i+1) = \frac{e^{-Y_i[l] f(x_i, l)}}{mk \prod_t Z_t} \quad (5)$$

Moreover, if $H(x_i, l) \neq Y_i[l]$ then $Y_i[l] f(x_i, l) \leq 0$ implying that $e^{-Y_i[l] f(x_i, l)} \geq 1$. Thus,

$$\delta(x_i, l) \leq e^{-Y_i[l] f(x_i, l)} \quad (6)$$

Combining Equations 4, 5, and 6, we get

$$WHL(L) = \frac{1}{mk} \sum_l w_l \sum_i \delta(x_i, l) \leq$$

$$\frac{1}{mk} \sum_l w_l \sum_i e^{-Y_i[l] f(x_i, l)} = \sum_l w_l \sum_i \left( \prod_{t=1}^{T} Z_t \right) D_{T+1}(i, l)$$

$$= \left( \prod_{t=1}^{T} Z_t \right) \sum_l w_l \sum_i D_{T+1}(i, l) \leq \left( \prod_{t=1}^{T} Z_t \right) \sum_l w_l$$

since $D_{T+1}(i, l) \leq 1$.

Define the constant $W = \sum_l w_l$.

Then

$$WHL(H) \leq \left( \prod_{t=1}^{T} W^{\frac{1}{t}} Z_t \right) \leq \left( \prod_{t=1}^{T} Y_t \right)$$

when $\hat{w}_l = w_l \times W^{\frac{1}{t}}$ ∎

Following the similar steps, this leads us a new criterion to select the weak hypotheses. Choose the weak learner which minimizes the following[1]:

[1]Note that using $\hat{w}_l = w_l$ is a reasonable approximation since $W^{\frac{1}{t}}$ converges to 1 quickly in the early iterations of the algorithm.

| | |
|---|---|
| Training Data Size | 9,094 utterances |
| Test Data Size | 5,171 utterances |
| Number of Call-Types | 84 |
| Call-Type Perplexity | 32.64 |
| Average Utterance Length | 10.66 words |

**Table 1**. Data characteristics used in the experiments.

$$Y_t = \sum_l \hat{w}_l \sum_j 2\sqrt{W_+^{jl} W_-^{jl}}$$

This corresponds to changing the selection criterion for the weak learner, $h_t$, in the AdaBoost algorithm. There is no change required in the algorithm presented in Figure 1.

## 4. EXPERIMENTS AND RESULTS

In order to evaluate our approach, we picked the task of call classification [9]. We carried out experiments using human-machine dialogs as collected by the AT&T natural spoken dialog system. We first describe this application, domain and data, and define the evaluation metrics. We then give the results obtained by the semantic classifier. We have performed our tests using the Boostexter tool [10]. For all experiments, we have used word $n$-grams as features and iterated 500 times.

### 4.1. Data

Our dataset includes human/machine dialogs collected from a customer care system. Table 1 summarizes the characteristics of our application including amount of training and test data, total number of call-types, average utterance length, and call-type perplexity. Perplexity is computed using the prior distribution over all the call-types in the training data.

### 4.2. Evaluation Metrics

Inspired by the information retrieval community, the classification performance is measured in terms of the *F-Measure* metric:

$$F - Measure = \frac{2 \times recall \times precision}{recall + precision}$$

where recall is defined as the proportion of all the true call-types that are correctly deduced by the classifier. It is obtained by dividing the number of true positives by the sum of true positives and false negatives. Precision is defined as the proportion of all the accepted call-types that are also true. It is obtained by dividing true positives by the sum of true positives and false positives. True (False) positives are the number of call-types for an utterance for which the deduced call-type has a confidence above a given threshold, hence accepted, and is (not) among the correct call-types. False (True) negatives are the number of call-types for an utterance for which the deduced call-type has a confidence

| | Overall | | | *Tellme(Balance)* | | |
|---|---|---|---|---|---|---|
| Weight | Recall | Precision | F-Measure | Recall | Precision | F-Measure |
| 1 | 0.57 | 0.77 | 0.656 | 0.66 | 0.87 | 0.750 |
| 100 | 0.57 | 0.77 | 0.659 | 0.76 | 0.88 | 0.813 |
| 10000 | 0.56 | 0.73 | 0.631 | 0.75 | 0.82 | 0.784 |

**Table 2**. Performance change when one calltype is weighted more than others.

| | Overall | | | Important Call-types | | |
|---|---|---|---|---|---|---|
| Weight | Recall | Precision | F-Measure | Recall | Precision | F-Measure |
| 1 | 0.57 | 0.77 | 0.656 | 0.49 | 0.73 | 0.575 |
| 100 | 0.57 | 0.76 | 0.649 | 0.53 | 0.77 | 0.610 |
| 10000 | 0.56 | 0.73 | 0.633 | 0.54 | 0.76 | 0.618 |

**Table 3**. Performance change when a set of calltypes is weighted more than others.

less than a threshold, hence rejected, and is (not) among the true call-types. The best F-Measure value is selected by scanning over all thresholds between 0 and 1.

### 4.3. Results

As a first experiment, we have chosen one moderately frequent calltype, namely *Tellme(Balance)*, occurring 189 times and increased its weight, while keeping all other weights as 1. Table 2 shows the change in the performance of that calltype and that of the overall. We have tried various weights. As seen, the F-Measure of that calltype has increased by 6.3% absolute without a significant change in the overall performance when the weight is set to 100. Note that when we continue increasing the weight, performance begins to deteriorate because of the decrease in the precision. Vast majority of the weak learners selected for the model trained with the weight of 10,000 are related to that calltype, making the other calltypes harder to win, and even results in worse performance for that calltype due to overtraining.

In order to see whether the same behavior is true for a set of important calltypes, not just one, we have randomly selected 12 calltypes, occurring 766 times, and gave them higher weights. Table 3 presents our results. The F-Measure for these calltypes increased by 3.5% absolute without a significant loss in overall performance with a weight of 100, but note the steady decrease in the overall performance with increasing weights.

## 5. CONCLUSIONS AND DISCUSSION

We have presented a cost-sensitive classification method, which extends the Boosting classifiers to allow the weighted classes. Our results indicate significant improvements in the "important" classes without a significant loss in the overall performance. Determining the weight for each class for optimum performance is determined by two major factors. The first one is the task or business needs. For example some classes can be considered to be more important than others for a specific task. The second one is the dynamic interaction of the weighted classes between each other. As seen in Table 2 overweighting a class may result in suboptimal performance. One solution to determine these weights would be beginning with a weight vector determined by business needs and then scaling them using a development test set.

**Acknowledgments** We would like to thank Robert E. Schapire for providing us the Boostexter classifier and Dilek Hakkani-Tür for many helpful discussions.

## 6. REFERENCES

[1] A. L. Gorin, G. Riccardi, and J. H. Wright, "Automated natural spoken dialog," *IEEE Computer Magazine*, vol. 35, no. 4, pp. 51–56, April 2002.

[2] P. Natarajan, R. Prasad, B. Suhm, and D. McCarthy, "Speech enabled natural language call routing: BBN call director," in *Proceedings of the ICSLP*, Denver, CO, September 2002.

[3] J. Chu-Carroll and B. Carpenter, "Vector-based natural language call routing," *Computational Linguistics*, vol. 25, no. 3, pp. 361–388, 1999.

[4] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, March 2001.

[5] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: Misclassification cost-sensitive boosting," in *Proceedings of the ICML*, Bled, Slovenia, June 1999.

[6] P. Domingos, "MetaCost: A general method for making classifiers cost sensitive," in *Proceedings of the KDD*, San Diego, CA, August 1999.

[7] B. Zadrozny, J. Langford, and N. Abe, "A simple method for cost sensitive learning," Tech. Rep., IBM, December 2002.

[8] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[9] G. Tur, J. Wright, A. Gorin, G. Riccardi, and D. Hakkani-Tür, "Improving spoken language understanding using word confusion networks," in *Proceedings of the ICSLP*, Denver, CO, September 2002.

[10] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.